

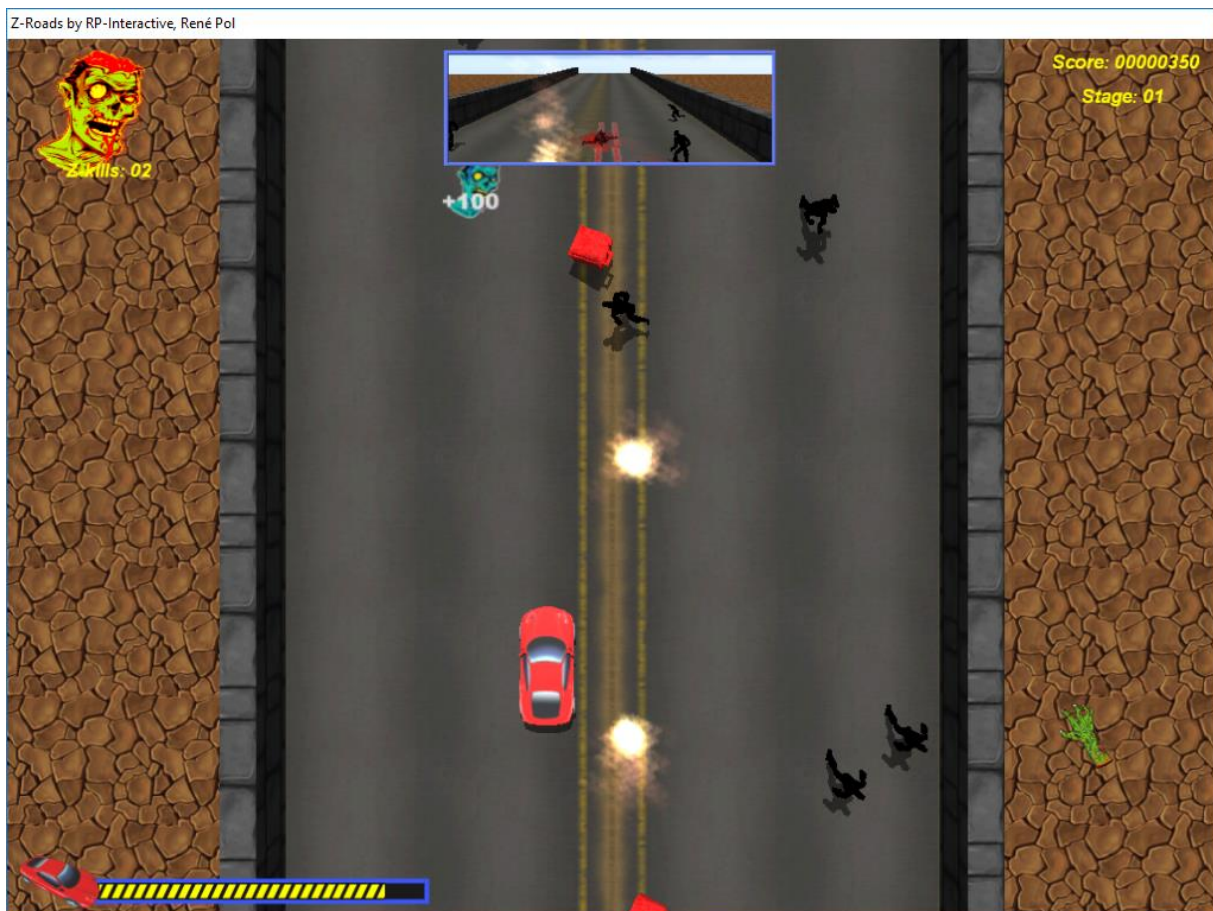


Game creation. It all starts all in the mind. After watching an episode of the Walking dead I came up with this idea for a simple game. The player drives a car from point a. to point b. While driving, it can run over zombies that stand or walk across the road in order to score points. Each time the player drives into something it is not supposed to its damage will go down. In this first part it means don't run through fire and leave those poor wolves alone. Getting to the end of the track with enough damage free? Then it's on to the next level. Got too much damage? Then its game over and the zombies will eat your brains.

In order to use this template you have to make yourself familiar with 3D Game studio A8 and it's easy to learn programming language Lite_C. As I am no professional coder I am sure some parts could be programmed better but the goal of this template is to provide you with a full working game that you can change to your own wishes by understanding all parts of the scripting (Programming).

I created al and you can use it freely as long as you give me some credits when this template helped you out. Know how to make things better ? Send me a PM in the forum or drop me an e-mail. You can reach me by using my site : <https://www.rp-interactive.nl>

From here I will walk you through each part of the script so you will understand which part does what in the game and how you can change it. Ready ? Let's begin.





Open the script and read along how all works.

Line 03-10.

All projects need their own work folder. In this work folder there should be subfolders that hold all stuff needed for the game. These lines are the so called paths. (They lead to these subfolder) Without these paths all the stuff the game uses would not be found.

```
#define PRAGMA_PATH "3Dmodels"  
#define PRAGMA_PATH "Graphics"  
#define PRAGMA_PATH "Scripts"  
#define PRAGMA_PATH "Sounds"
```

In this example I created folders for :3D models, Graphics, Scripts and sounds.

Line 12-49.

Here you find all variables that are used. A Variable is like a container in which you can store stuff and get stuff out by calling the variable in script (Programming)

A variable that has no panel (Graphic not needed) Is always noted like this :

```
var (name of the variable);
```

If you need the variable to start at a certain number it is always noted like this :

```
var(name of the variable) = Number the variable must be set on.
```

If you need the variable to be visible during game you have to create a panel for it. Here is how :

First the name of the variable :

```
var score = 0;
```

Then we create a font that the variable should use. In this case the font is defined by fnt3_pan and it uses Ariel Black at size 18 in bold(B) and Italic(I) style.

```
FONT* fnt3_pan = "Ariel Black#18bi";
```

Now give the panel a name and give it the position where it needs to be shown on screen. (X and Y axis). In this line we also state the name of the font it must use. The %08.0f line means it should display 8 numbers. Change it into any amount of numbers you see fit.

```
PANEL* pan_score = {digits=885,10,"Score: %08.0f",fnt3_pan,1,score;
```



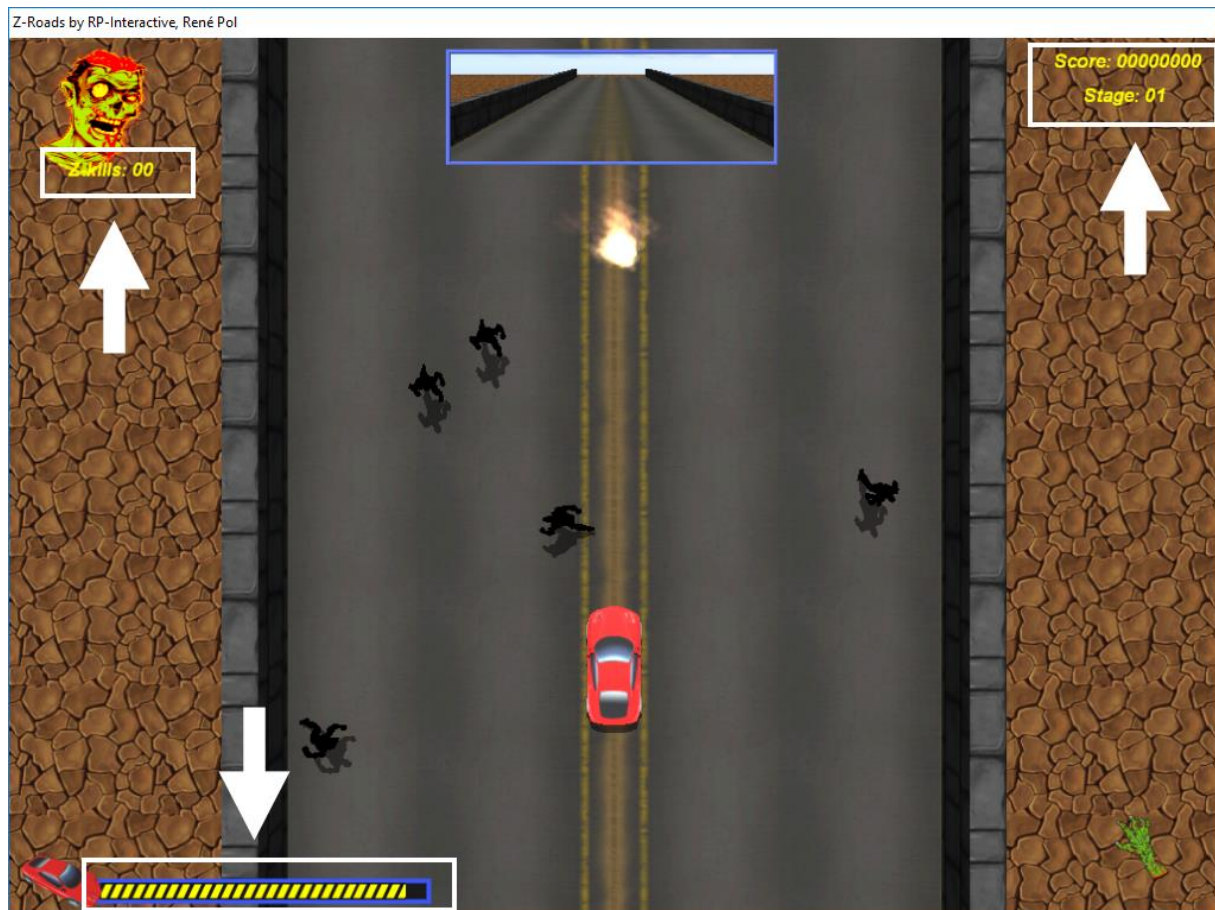
Next we give it a layer. In this case layer 10. If another panel has a higher layer number it will be placed on top of this one. If lower it will be placed under this one.

```
layer = 10;
```

At the end we place the so called fags. SHOW means it must be visible during game. Then we see a colour code (In this case yellow) change the numbers to get the colours you want to use.

```
flags = SHOW;green=255; blue=0; red=255;
```

All clear to you? Good then you know how to create variables and how to make them visible and set them on any starting value.





Line 51-58.

In order to create an outdoor look you can use a skybox picture. This picture will be folded by script around your level so it will give your level an outdoor look.

This is the skybox that is used.



We define the sky as an entity that is used :

```
ENTITY* sky =
```

Then we define the picture it is using for the sky:

```
type = "skysunset+6.tga";
```

As flags we set Sky use it as cube (Fold it) and make it visible (SHOW). The picture should use always be made in the power of 2. (flags2)

```
flags2 = SKY | CUBE | SHOW;
```

So this simple piece of script will create a nice looking sky around your level.





Line 62.

To give an entity an unique name we use a pointer. This way you can call the pointer by script whenever needed. Don't worry you will see this later in the script.

You define a pointer by creating this line:

```
ENTITY* zcar;
```

This pointer is created for the payer's car model and it's called zcar.

Line 67-74.

Every sound used must be defined first in order to be called later in the script. Defining a sound always is noted as :

```
SOUND* ding_snd (Name sound) = "Name of to use wav/mp3/ogg";
```

Do this for all sounds that you are planning to use so they can be called up anywhere in the script.

Line 79-212.

All panels (graphics) that are used must be defined first. A simple panel is always noted like this:

```
BMAP* menu_map (name of the panel = "Picture name";
```

Then we define the panel and the map it must use.

```
PANEL* menu_pan =  
bmap = menu_map;
```

Set the position on the X and Y axis. (Where to set it on screen)

```
pos_x = 0;  
pos_y = 0;
```

If you want a button to be displayed on the panel make sure you have 3 button pictures. One for on, Second for off, Third for over. Start the script line with button. Then the position on x and y axis followed by the 3 pictures used for the button. Then you set the function it should use (Function name) followed by Null, Null. So when on button is used it will start any given function. Over and off are not used so set to Null.

```
button (10, 600, "playbut02.png", "playbut01.png", "playbut01.png", begin_game, NULL, NULL);
```

Then set the layer and flags. You should know what they are now.

```
layer = 21;  
flags = SHOW;
```




Simple right ? Now let me explain how to create a so called health bar. In this game I use it as a damage bar. (To see the damage in a graphic)

First we define the bar picture we use like we learned before and set the panel.

```
BMAP* test_map = "bar.png";  
PANEL* back_pan =
```

We define the bar picture, set its position on the x and y position. The bar picture is 279pixels width so we set it on full and call it damage as it uses the variable damage.

```
hbar = 78, 739, 279, test_map, 1, damage;
```

End it by setting a layer and setting the flags to use.

```
layer = 5;  
flags = VISIBLE | OVERLAY;
```

Now you know how to use panels (Graphics) Health bars and buttons.



This way we create all panels used. So also the game over, pause, scoring panels and more.



line 225-246.

In this game we have a rear view mirror that shows the view from behind the player. Here is how to make this view.

We create a panel as we have learned for the mirror.

```
BMAP* mirror_map = "mirror.png";  
PANEL* mirror_pan =  
bmap = mirror_map;  
pos_x = 370;  
pos_y = 10;  
layer = 15;  
flags = SHOW;
```

Then we create a view with the VIEW command.*

```
VIEW* new_view =  
pos_x = 370;  
pos_y = 10;  
size_x = 280;  
size_y = 100;  
arc = 60;  
layer = 4;  
flags = SHOW;
```

This is all. Later in the script we call this view up in the player script. This way we can create multiple views in a game but use it also to create split screen games.





line 250-253.

In order to make functions or actions work that are scripted lower in the script we need to preload them first so the functions and actions can be called up from anywhere in the script.

Basically it's a line that is the name of the function closed with a ;

```
function begin_game();
```

line 258-296.

This script is made by 3Run and It will create a nice fire to any object you want.

Study it and change some numbers and colours to make it your own. Change the life spawn or size and toy with this script.

```
var start_color[3]; <<<<< start color of the particle
var end_color[3]; <<<<<end color of the particle
var current_color[3]; <<<<<current color of the particle
var flame_lifespan = 10;<<<<duration of the particle
```

```
function vec_randomize(VECTOR* vec, var range){<<<<< set position at random for particles.
vec_set(vec, vector(random(1) - 0.5, random(1) - 0.5, random(1) - 0.5));
vec_normalize(vec, random(range));
}
```

```
function flame_alphafade(PARTICLE* p){ <<<<<name of function
```

```
start_color[0] = 200;
start_color[1] = 180;
start_color[2] = 100;
end_color[0] = 128;
end_color[1] = 50;
end_color[2] = 50;
vec_lerp(current_color[0], end_color[0], start_color[0], p.lifespan / flame_lifespan);
p.red = current_color[0];
p.green = current_color[1];
p.blue = current_color[2];
p.alpha = p.lifespan / flame_lifespan * 100 * 0.3;
if(p.alpha <= 0){ p.lifespan = 0; }
}
```

```
function effect_flame(PARTICLE* p){<<<<< name of function
VECTOR vTemp;
vec_randomize(vTemp, 4 + random(1));
vec_add(p.vel_x, vTemp);
p.size = 15;
p.bmap = flame_tga;
p.gravity = -3;
p.lifespan = flame_lifespan;
set(p, MOVE | BRIGHT | TRANSLUCENT);
p.event = flame_alphafade;
}
```




Line 299-318.

In this game we use an explosion animated sprite. In order to make it show the animation once we use this script for playing it.

```
function sprite_played() <<<<< The name of the function
{
set (my, PASSABLE | TRANSLUCENT); <<<<< set the sprite passable and translucent
my.scale_x = 0.5; <<<<< scale it down a bit
my.scale_y = my.scale_x; <<<<< scale on x and y axis
my.ambient = 100; <<<<<<< set the ambient on 100
my.roll = random(360); <<<<< make it roll randomly between 0 and 360 degrees
my.alpha = 100; <<<<<<< set the translucent alpha on 100 so it shows
while (my.frame < 16) <<<<< while the animation frame is under (Smaller) then 16
{
my.frame += 1 * time_step; <<<<< show next animation frame 1 + 1 and so on.
wait (1); <<<<< wait one frame
}
while (my.alpha > 0) <<<<< while the alpha (Translucent) is bigger then 0
{
my.alpha -= 5 * time_step; <<<<< Make alpha go down on speed 5
wait (1); <<<<<< wait one frame
}
ent_remove (me); <<<<<< remove the complete sprite after the animation
}
```





Line 324-357.

When the player reaches the end of the level it will do some scoring and goes to the next level. That's why I created this level switch function. It will be called whenever needed.

```
function level_switch() <<<<< Name of the function
{
set(black_pan,SHOW | TRANSLUCENT);<<<<< set the black panel and its translucent
black_pan.alpha = 0; <<<<<<set the alpha on 0 so it's not visible
while(black_pan.alpha <100){ <<<<<<while the alpha is smaller then 100
black_pan.alpha +=5*time_step; <<<<< make the black panel fade in on a speed of 5
wait(1); <<<<<<wait one frame
}
reset(pan_zbonus,SHOW);<<<<<reset the bonus panel (Make it invisible)
reset(bonus_pan,SHOW);<<<<<<reset the bonus panel(Make it invisible)
zbonus =0; <<<<<se the bonus variable to 0
stage +=1;<<<<<< ad one to the stage variable
wait(-1);<<<<<<hold for one minus frame
}
if(stage ==2){<<<<<<<if the stage variable is equal to 2
damage=279;<<<<<< restore the damage variable to 279
game_enable = 0; <<<<<<set the game_enable variable to 0
level_load("stage002.WMB"); <<<<<< load level 2
ent_create("zcar01.mdl", vector(-150,0,-5), zombie_car);<<<<<< create the player car model at level load
}
while(black_pan.alpha >0){<<<<<<While the alpha of black is bigger then 0
black_pan.alpha -=5*time_step; <<<<<< make it fade out on speed 5
wait(1);<<<<<<< wait one frame
}
begin_newlevel();<<<<<<<call function begin_newlevel
}
```

You see you already use all stuff we scripted before. We use the variables, panels and earlier made functions. All script is connected. When stage 3 is reached you just copy and insert lines 343-347 and make it load stage003 and so on. There you have it a simple level changing script.

Line 362-379.

To give this game an arcade feeling I created a function that makes a score picture show up that floats of the screen. I saw this style first in a demo game created by 3Run and used it ever since ☺

```
function score_up() <<<<< Name of the function
{
my.tilt =90;<<<<< make it tilt 90 degree
my.pan= 180;<<<<< set direction of sprite 180 degree
my.alpha = 100; <<<<<<< set its alpha (Translucent) to 100
set(my,PASSABLE | TRANSLUCENT);<<<<<< Make it passable and use the Translucent
while(1)<<<<<<<<while on screen
{
my.alpha -=1*time_step; <<<<<< make it fade out on speed 1
my.x +=40*time_step;<<<<<<< move on x axis on speed 40
my.z +=1*time_step;<<<<<<< make it move on axis on speed 1
wait(1);<<<<<<<<wait one frame
if(my.alpha <=0){<<<<<<<<when alpha is smaller or equal to 0
ent_remove(me);<<<<<<<<remove the sprite
break;<<<<<<<<break off the full function
}
}
```



Line 388-406.

*Every time a new level starts we make it use the countdown before the player starts to drive.
So when the level_switch function has been activated the next script part will begin.*

```
function begin_newlevel() <<<<< name of the function
{
wait(-1);<<<<< wait one minus frame
snd_play(carengine_snd,150,0);<<<<< play the car engine start sound
set(three_pan,SHOW);<<<<< make panel three visible
wait(-1);<<<<< wait one minus frame
reset(three_pan,SHOW);<<<<< make panel three invisible
set(two_pan,SHOW);<<<<<< make panel two visible
wait(-1);<<<<< wait one minus frame
reset(two_pan,SHOW);<<<<< make panel two invisible
set(one_pan,SHOW);<<<<< make panel one visible
wait(-1);<<<<< wait one minus frame
reset(one_pan,SHOW);<<<<< make panel one invisible
set(go_pan,SHOW);<<<<< make panel go visible
game_enable =1;<<<<<< set game_enable variable to 1
wait(-1);<<<<< wait one minus frame
reset(go_pan,SHOW);<<<<<< make panel go invisible
}
```

*See what happens here ? The panels we defined are now used and they get shown in our given order.
We make them disappear and we change the variable we created earlier. The wait(-1) makes a sort of
short pause before the next line starts doing its thing. So here you learned how to use panels
(Graphics) and make them visible or invisible. Also you learned how to make the sound play that you
defined earlier in the script.*



Line 411-442.

When we start a complete new game we need to reset stuff and reload the first level. Here is the function that just does that for us.

```
function begin_game()<<<< function name
{
mouse_mode= 0;<<<< make mouse cursor invisible
set(black_pan,SHOW | TRANSLUCENT);<<<< make panel black show and use translucent
black_pan.alpha = 0;<<<<set alpha on 0
while(black_pan.alpha <100){<<<<when alpha is smaller then 100
black_pan.alpha +=5*time_step; <<<< make panel show fade in on speed 5
wait(1); <<<<< wait one frame
}
wait(-1);<<<<<wait one frame
ent_create("zcar01.mdl", vector(-150,0,-5), zombie_car);<< create the car model at x,y, coordinates in level and give it the ombie car action
reset(menu_pan,SHOW);<<<<<<< reset the menu panel (Make it invisible)
while(black_pan.alpha >0){<<<<<when alpha is bigger then 0
black_pan.alpha -=5*time_step; <<<<< make alpha fade out on speed 5
wait(1);<<<<< wait one frame
}
reset(black_pan,SHOW | TRANSLUCENT);<<<<reset the black panel
wait(-1);<<<< wait minus one frame
snd_play(carengine_snd,150,0);<<<<<play the car engine sound
```

Then the countdown part that is also used in begin_new_level function so no need to explain again right ?

```
set(three_pan,SHOW);
wait(-1);
reset(three_pan,SHOW);
set(two_pan,SHOW);
wait(-1);
reset(two_pan,SHOW);
set(one_pan,SHOW);
wait(-1);
reset(one_pan,SHOW);
set(go_pan,SHOW);
game_enable =1;
wait(-1);
reset(go_pan,SHOW);
}
```

Line 448-458.

When the player reaches the end of the level the player car gets replaced by the same model but with a different action (It will drive off screen) This is the action.

```
action race_out() <<<< name of the action
{
scoring_now(); <<<<< start the function scoring_now
set(my,SHADOW | METAL | PASSABLE); <<<<< Give the model shadow, metal look and make it passable
my.pan = 0; <<<< set the models direction straight forward.
while(1) <<<<< while on screen
{
c_move(my, vector(35* time_step, 0, 0), nullvector, GLIDE | IGNORE_SPRITES | IGNORE_PASSABLE);<<<<< movement instruction from
point of origin move on x axis with speed of 35 no y and movement (0) glide along walls and ignore sprites and passable objects.
wait(1);<<<<<<wait one frame
}
```



Did you notice that in this script you called another part of the script ? That's right the `scoring_now` function. You learned the `c_move` is used when it comes to programming movement scripts. Now you could make the car disappear when its off screen but why would we ? A new level will be loaded using the `level_switch` function so the car will be removed automatically.

Line 462-469.

While scoring we make a sound repeat until the scoring is done.

```
function play_sound()<<<< name of function
{
while(sound_on ==0) <<<< when the variable sound_on is equal to 0
{
snd_play(ding_snd,150,0); <<<<< play sound
wait(-0.1);<<<<wait minus one frame
}
```

The sound will play in the while loop until the variable `sound_on` is set different from 0.

Line 473-493.

I created a way of scoring so it counts the zombie kill variable, multiplies it with 100 and then adds it to the score variable. During scoring the sound will play with the function we just created.

```
function scoring_now()<<<< name of the function
{
sound_on = 0;<<<<< set sound_on variable to 0
set(bonus_pan,SHOW);<<<<< make bonus panel show
set(pan_zbonus,SHOW);<<<<< make zombie bonus panel show
play_sound();<<<<< start the play_sound function
while(1)<<<<<<while scoring now is activated
{
zkills -=0.5*time_step;<<<<<<subtract kills bonus variable on 0.5 speed
zbonus +=0.5*time_step;<<<<<<add bonus variable on 0.5 speed
wait(1);<<<<<<wait one frame
if(zkills <0){<<<<< if kills variable is smaller than 0
zkills =0; <<<<< set kills variable to 0
sound_on =1; <<<<< set sound on variable to 1
score = zbonus*100 +score;<<<<< set score variable as bonus variable multiplied by 100 + score variable
wait(-5);<<<< wait minus 5 frames
level_switch();<<<<<<start the level_switch function
break;<<<<< Break off the function
}
```





Line 497-512.

To pause the game I created a pause_game function. On the press of the P key it will pause the game and on press again it will continue the game. It also pauses the music that's why the in game song uses a variable called soundtrack_handle. This variable we created earlier in the script.

```
function pause_game()<<<< Name of the function
{
if(freeze_mode == 2) <<<< if freeze _mode is equal to 2
{
freeze_mode = 0; <<<< set freeze _mode to 0
reset(break_pan,SHOW | TRANSLUCENT);<<<<<reset the pause panel
media_start(soundtrack_handle);<<<<<restart the game music
}
else<<<<<if none of above then
{
freeze_mode = 2; <<<<when freeze _mode is 2
set(break_pan,SHOW | TRANSLUCENT); <<<<<set the pause panel
media_pause(soundtrack_handle);<<<<<pause the in game music
}
}
```





Line 517-547.

Every project has a so called main function. This function holds all that is needed to start up the level and to activate needed functions. You can set the resolution and if the game should run full screen or in a window. Make sure your project has a main function without it your game would do nothing.

```
function main()<<<<< function name
{
mouse_pointer=0;<<<<<makes windows mouse cursor invisible in game screen
shadow_stencil = 2; <<<<< Use stencil shadows (Real shape shadow)
set(black_pan,SHOW|TRANSLUCENT);<<<<< Set the black panel and us translucent
set(logo_pan,SHOW);<<<<< set the logo panel
video_set(1024,768,32,0);<<<< set resolution to 1024x768 with 32 bit colour and start in window 0
video_window(vector(0,0,0),vector(1024,768,0), 16, "Z-Roads by RP-Interactive, René Pol");<<< give window a start position and title bar
level_load("Z-roads.WMB");<<<<< load the level
black_pan.alpha = 100;<<<<< set black alpha on 100 so fully visible
while(black_pan.alpha >0){<<<<<when black alpha is bigger then 0
black_pan.alpha -=3*time_step;<<<< fade black out at speed 3
wait(1);<<<< wait one frame
}
wait(-1);<<<< wait one minus frame
soundtrack_handle = media_loop("ingame.mp3", NULL, 200);<<<start the in game music in a loop using the variable soundtrack_handle
while(black_pan.alpha <100){<<<<<while black alpha is smaller then 100
black_pan.alpha +=3*time_step;>>>>>> fade black in with speed 3
wait(1);<<<<<wait one frame
}
wait(-1);<<<<<wait one minus frame
reset(logo_pan,SHOW);<<<<<make logo panel disappear
mouse_map = cursor_png; <<<<<use defined picture as mouse cursor
mouse_mode =3;<<<<< set mouse mode to 3
while(black_pan.alpha >0){<<<<<if black alpha is bigger then 0
black_pan.alpha -=3*time_step; <<<< fade black alpha out on speed 3
wait(1); <<<<< wait one frame
}
reset(black_pan,SHOW|TRANSLUCENT);<<<<<reset black pan make it invisible
on_p = pause_game;<<<<< on p press the pause_game function will be calles at any time
}
```

If you payed attention you see we use the fading panels technique a lot. It gives the game a nice visual touch. So in this main function we make the music play in a loop using media_loop and the variable we created for the song (soundtrack_handle). We create a mouse cursor at start up and the pause function. Both functions we created earlier in the script.



Line 552-575.

When our player has too much damage it will be game over so it needs a game over function.

```
function game_over()<<<<< function name
{
wait(-5);<<<<< wait minus 5 frames
set(black_pan,SHOW | TRANSLUCENT);<<<<< show black panel
black_pan.alpha = 0;<<<<< set the alpha to 0
while(black_pan.alpha <100){<<<<<when alpha is smaller then 100
black_pan.alpha +=5*time_step;<<< fade black in at speed 5
wait(1);<<<<< wait one frame
}
wait(-1);<<<<< wait one minus frame
mouse_mode =3;
set(menu_pan,SHOW);<<<<< set the menu panel visible
game_enable = 0; <<<<<set the game enable variable to 0
level_load("Z-roads.WMB");<<<<< load the begin level
score = 0;<<<<<set score variable back to 0
zkills = 0;<<<<<set skills variable back to 0
damage=279;<<<<<restore damage to 279
reset(ga_pan,SHOW);<<<<<reset the game over panel make it invisible
while(black_pan.alpha >0){<<<<<if alpha is bigger then 0
black_pan.alpha -=5*time_step;<<<<< fade out black at speed 5
wait(1);<<<<<wait one frame
}
reset(black_pan,SHOW |TRANSLUCENT);<<<<<reset the black pan complete.
}
```





Line 580-591.

When the play has too much damage its game over. The car model will be replaced with an upside down car model that is on fire.

```
action exploded()<<<< action name
{
set(my,SHADOW | METAL);<<<< give it shadow and metal look
snd_play(explo2_snd,150,0);<<<< play explosion sound
game_over();<<<< start the game over function
ent_create("Explosion+16.tga", vector(my.x,my.y,my.z), sprite_played);<<<<create the explosion sprite that uses animation script and
create it at the models coordinates.

while(1)<<<< when this is on screen
{
effect(effect_flame, 1, my.x, nullvector);<<<< use the fire script and attach it to the model
wait(1);<<<<wait one frame
}
}
```

Line 597-667.

Now an important script. The movement of the player and its camera view. In this game we use a car but imagine a space ship moving the same way or a character, you can make any kind of scrolling game with this script. The rear view mirror script is used so we can see what is happening behind the player.(VIEW script)

```
action zombie_car()<<<< name of action
{
set(my,SHADOW | METAL);<<< give model shadow and metal look
zcar = me;<<<< use the pointer we created so the model has a unique name
var speed = 25;<<<< create a speed variable and set it on 25
c_setminmax(me);<<<<use the models real sie for collision
while(1)<<<<when on screen
{
while(game_enable ==1 && damage >0)<<<<when the game enable variable is equal to 1 and damage variable is bigger then 0
{
if(key_cul){<<<<when cursor left is pushed
my.pan = 6;<<<< set direction 6 degree
}
if(key_cur){<<<<when cursor right is pushed
my.pan = -6;<<<<set direction -6 degree
}
if(!key_cur && !key_cul){<<<<when other key is pressed then cursor left and right
my.pan = 0;<<<<set direction to 0
}
damage -=0.6*time_step;<<<<make damage variable go down at speed 0.6
my.y += 25 * (key_cul - key_cur) * time_step;<<<<<move on y axis with the cursor left and right keys
my.y = clamp(my.y,-220,215);<<<<< give the model boundaries so it can't go off the road.
c_move(my, vector(speed*time_step, 0, 0), nullvector, GLIDE | IGNORE_SPRITES | IGNORE_PASSABLE); <<<< use c_move instruction for
movement from models origin. Move on variable speed on x axis no movement on y and ignore all sprites and passable models.

camera.tilt = -90;<<<< set camera view -90 degree (Straight down)
camera.x = zcar.x+150;<<<<< set camera x in front of model pointer x axis
camera.z = zcar.z + 800;<<<<< set camera. On axis using pointer
new_view.tilt = -10;<<<<< using the view we scripted earlier set tilt on -10 degree
new_view.pan = -180; 10;<<<<< using the view we scripted earlier set pan on -180 degree
new_view.y = zcar.y;<<<< connect new view y to the pointer y
new_view.x =zcar.x-80;<<<<connect new view x to the pointer x - 80
new_view.z = zcar.z+130;<<<<<<connect new view to pointer + 130
wait (1); <<<< wait one frame
}
```



```
if(damage <=40){<<<<if damage variable is smaller or equal to 40
effect(effect_flame, 1, my.x, nullvector);<<<< attach the fire script and make it stick to the model
}
if(my.x >19441.801){<<<<<if the x axis distance in level is bigger then 19441.801
set(zcar,INVISIBLE | PASSABLE);<<<<<set pointer invisible
reset(zcar,SHADOW);<<<<<reset the pointers shadow
ent_create("zcar01.mdl", vector(zcar.x,zcar.y,zcar.z),race_out);<<<<< create a new model at pointers position and give it the race out
action
wait(1);<<<<<wait one frame
ent_remove(zcar);<<<<< remove the pointer model
return;<<<<<go back to beginning action.
}
if(damage <1){<<<<<<when damage is smaller than 1
set(my,INVISIBLE | PASSABLE);<<<<<<set pointer invisible
reset(my,SHADOW);<<<<<<reset pointers shadow
snd_play(explo2_snd,150,0);<<<<<<play explosion sound
ent_create("Explosion+16.tga", vector(zcar.x,zcar.y,zcar.z), sprite_played);<<<<<<create explosion sprite at pointer position
ent_create("dcar.mdl", vector(zcar.x,zcar.y,zcar.z),exploded);<<<<<<Create new model at pointer position with exploded action
wait(1);<<<<<< Wait one frame
ent_remove(me);<<<<<<remove the pointer model
set(ga_pan,SHOW);<<<<<<set the game over panel
break;<<<<<<break off the action
}
if(game_enable ==0)<<<<<<when game_enable variable is equal to 0
{
new_view.tilt = -10;<<<<<< using the view we scripted earlier set tilt on -10 degree
new_view.pan = -180; 10;<<<<<< using the view we scripted earlier set pan on -180 degree
new_view.y = zcar.y;<<<<< connect new view y to the pointer y
new_view.x = zcar.x-80;<<<<<connect new view x to the pointer x - 80
new_view.z = zcar.z+130;<<<<<<connect new view to pointer + 130
camera.tilt = -90;<<<<<< set camera view -90 degree (Straight down)
camera.x = zcar.x+150;<<<<<< set camera x in front of model pointer x axis
camera.z = zcar.z + 800;<<<<<< set camera. On axis using pointer
wait (1);<<<<<< wait one frame
}
}
```

There you have it. A working player movement code. If you take a good look you see it uses functions ,actions, variable and panels we scripted earlier. A lot of scripts are the same with little changes made to them.

Line 673-687.

When a zombie is run over a dead zombie will appear that fades out. Here is the script it uses :

```
function d_zombie()<<<<<< name of function
{
set(my,TRANSLUCENT | PASSABLE);<<<<<<set model translucent and passable
my.alpha = 100;<<<<<<set models alpha on 100 (Visible)
my.pan = integer(random(180));<<<<<< give random direction between 0 and 180 degree
while(1)<<<<<< when on screen
{
my.alpha -=2*time_step;<<<<<<make model fade out at speed 2
wait(1);<<<<<< wait one frame
if(my.alpha <=0){<<<<<<if alpha is smaller or equal to 0
ent_remove(me);<<<<<<remove the model
break;<<<<<< break off the function
}
}
```




You should by now understand how to fade in and out models and panels as we used it many times and will be using it some more in the rest of the script. Its al about using the alpha and the flag TRANSLUCENT.

Line 691-707.

While hitting those nasty walkers blood puddles will appear and we use basically a same script for them.

```
function blood_pool()<<<< name of the function
{
  snd_play(zdead_snd,50,0);<<<<play sound
  set(my,TRANSLUCENT | PASSABLE);<<<<Set sprite translucent and passable
  my.alpha = 100;<<<<set alpha to 100 (Visible)
  my.tilt = 90;<<<<tilt the sprite 90 degree
  my.pan = integer(random(180));<<<<set random direction between 0 and 180
  while(1)<<<<when on screen
  {
    my.alpha -=1*time_step;<<<< fade out at speed of 1
    wait(1);<<<<< wait one frame
    if(my.alpha <=0){<<<<< when alpha is smaller or equal to 0
    ent_remove(me);<<<<<remove the sprite
    break;<<<<< break off the function
  }
}
```

Line 711-727.

Bloody tracks will appear while hitting zombies. It uses again the same kind of script as the blood and dead zombie. So you see a lot of stuff is scripted in the same way.

```
function track()<<<< name of the function
{
  set(my,TRANSLUCENT | PASSABLE);<<<<< set sprite translucent and passable
  my.alpha = 100;<<<<<set alpha to 100
  my.tilt = 90;<<<<tilt the sprite 90 degree
  my.pan = zcar.pan;<<<<use the same direction as the pointer
  while(1)<<<<when on screen
  {
    my.alpha -=1*time_step;<<<<< fade out at speed 1
    wait(1);<<<<<<wait one frame
    if(my.alpha <=0){<<<<<<if alpha is smaller or equal to 0
    ent_remove(me);<<<<<remove sprite
    break;<<<<<break off function
  }
}
```



Line 734-803.

I created 2 kinds of zombies. Standing zombies and walking zombies. They each react different on impact and use in script the so called emask. First we create the impact functions for the 2 different zombie actions.

Function for the standing zombie

```
function zombie_impact()<<<<< function name
{
snd_play(carhit_snd,40,0);<<<<< play sound
score+=100;<<<<< add 100 to score variable
set(zombieh_pan,SHOW);<<<<< set panel visible
zkills +=1;<<<<<add 1 to kills variable
set(my,INVISIBLE | PASSABLE);<<<<<set model invisible and passable
reset(my,SHADOW);<<<<<reset the models shadow
ent_create("hundred.png", vector(my.x,my.y,my.z+50), score_up);<<<<<create score panel with score up function at models coordinates.
ent_create("blood.png", vector(my.x,my.y,my.z-18), blood_pool);<<<<<create blood and bloodpool function at models coordinates.
ent_create("dzombie.mdl", vector(my.x,my.y,my.z-14), d_zombie);<<<<<create new model with d-ombie action at models coordinates.
wait(-0.3);<<<<<wait minus 3 frames
snd_play(blood_snd,50,0);<<<<<play sound
ent_create("track.png", vector(zcar.x-50,zcar.y+10,zcar.z), track);<<<<< create track at pointer coordinates
ent_create("track.png", vector(zcar.x-50,zcar.y-15,zcar.z), track);<<<<< create track at pointer coordinates
wait(1);<<<<< wait one frame
ent_remove(me);<<<<< remove the model
wait(-0.2);<<<<< wait minus 2 frames
reset(zombieh_pan,SHOW);<<<<< reset panel make it invisible
}
```

For the walking zombie we need separate parts as it needs to turn on block collision and only dies on impact or entity emask.

```
function zombie_impact2()<<<<<name of the function
{
if (event_type == EVENT_ENTITY)<<<<< if emask event is entity
{
snd_play(carhit_snd,80,0);<<<<<play sound
ent_create("hundredf.png", vector(my.x,my.y,my.z+50), score_up); <<<<<create score panel with score up function at models coordinates.
score+=250;<<<<< add 250 to score variable
set(zombieh_pan,SHOW);<<<<< make panel visible
zkills +=1;<<<<< add 1 to kills variable
set(my,INVISIBLE | PASSABLE);<<<<<set model invisible and passable
reset(my,SHADOW);<<<<<reset models shadow
ent_create("blood.png", vector(my.x,my.y,my.z-18), blood_pool);<<<<<create blood and bloodpool function at models coordinates
ent_create("dzombie.mdl", vector(my.x,my.y,my.z-14), d_zombie); <<<<<create new model with d-zombie action at models coordinates.
wait(-0.3);<<<<<wait minus 0.3 frames
snd_play(blood_snd,50,0);<<<<< play sound
ent_create("track.png", vector(zcar.x-50,zcar.y+10,zcar.z), track);<<<<< create track at pointer coordinates
ent_create("track.png", vector(zcar.x-50,zcar.y-15,zcar.z), track);<<<<< create track at pointer coordinates
wait(1);<<<<< wait one frame
ent_remove(me);<<<<< remove model
wait(-0.2);<<<<<wait minus 0.2 frames
reset(zombieh_pan,SHOW);<<<<<make panel invisible
}
```



```
if (event_type == EVENT_IMPACT) <<< if emask event is impact
{
    snd_play(carhit_snd,80,0);<<<<<play sound
    ent_create("hundredf.png", vector(my.x,my.y,my.z+50), score_up); <<<<<create score panel with score up function at models coordinates.
    score+=250;<<<< add 250 to score variable
    set(zombieh_pan,SHOW);<<<<<< make panel visible
    zkills +=1;<<<<< add 1 to kills variable
    set(my,INVISIBLE | PASSABLE);<<<<set model invisible and passable
    reset(my,SHADOW);<<<<<reset models shadow
    ent_create("blood.png", vector(my.x,my.y,my.z-18), blood_pool);<<<<<create blood and bloodpool function at models coordinates
    ent_create("dzombie.mdl", vector(my.x,my.y,my.z-14), d_zombie); <<<<<create new model with d-ombie action at models coordinates.
    wait(-0.3);<<<<<wait minus 0.3 frames
    snd_play(blood_snd,50,0);<<<<< play sound
    ent_create("track.png", vector(zcar.x-50,zcar.y+10,zcar.z), track);<<<< create track at pointer coordinates
    ent_create("track.png", vector(zcar.x-50,zcar.y-15,zcar.z), track);<<<< create track at pointer coordinates
    wait(1);<<<<<< wait one frame
    ent_remove(me);<<<<< remove model
    wait(-0.2);<<<<<wait minus 0.2 frames
    reset(zombieh_pan,SHOW);<<<<<make panel invisible
}
if (event_type == EVENT_BLOCK)<<<<<if emask event is block
{
    my.pan +=180;<<<<<set direction 180;
}
}
```

Line 808-855.

Time to create the 2 actions for the zombies. They each use an emask that will call the function (event) that is connected to it. So on hit it activates the event.

```
action standing_zombie()<<<<< name action
{
    set(my,SHADOW);<<<< give model shadow
    var stand_percentage;<<<< create a variable for standing animation
    c_setminmax(me);<<<<<use real sie for collision
    my.emask |= ENABLE_ENTITY | ENABLE_IMPACT;<<<<<set the emask on entity and impact
    my.event= zombie_impact;<<<<create an event that starts the function ombie_impact.
    while(!zcar){wait(1);<<<<<<when the pointer is not in game just wait
    }
    while(1)<<<<<when pointer is in game
    {
        stand_percentage %= 100;<<<<<use 100 procent of animation variable
        stand_percentage += 1 * time_step;<<<<<go through animation at speed 1
        ent_animate(my, "stand", stand_percentage, ANM_CYCLE);<<<<<use models standing animation and keep repeating it (ANM_CYCLE)
        wait(1);<<<<<wait one frame
        if (my.x < zcar.x-2000){<<<< when the model is behind the pointer 2000 or more
            ent_remove(me);<<<<<remove the model
            break;<<<<< break of the action
        }
    }
}
```



```
action walking_zombie()<<<< name of action
{
set(my,SHADOW);<<<< give model shadow
var walking_percentage;<<<<create a variable for walking animation
c_setminmax(me);<<<<use real sie for collision
my.emask |= ENABLE_ENTITY | ENABLE_IMPACT | ENABLE_BLOCK; <<<<set the emask on entity and impact and block
my.event= zombie_impact2; <<<<create an event that starts the function zombie_impact.
while(!zcar){wait(1);<<<< when the pointer is not in game just wait
}
while(1)<<<<while pointer is in game
{
walking_percentage %= 100; <<<<use 100 percent of animation variable
walking_percentage += 2 * time_step; <<<<go through animation at speed 1
ent_animate(my, "run", walking_percentage, ANM_CYCLE); <<<<use models walking animation and keep repeating it (ANM_CYCLE)
c_move(my, vector(1* time_step, 0, 0), nullvector, GLIDE | IGNORE_SPRITES | IGNORE_PASSABLE);<<<< make model move from origin at
speed 1 on x axis no movement on y and . Use glide and ignore sprites and passable models.
wait(1);<<<<wait one frame
if (my.x < zcar.x-2000){<<<< when the model is behind the pointer 2000 or more
ent_remove(me);<<<<remove the model
break;<<<< break of the action
}
}
}
```

Line 860-882.

In this game we have fire objects. When run into them they will do damage to the player. You can place fire object anywhere you want.

```
action fire_object()<<<< name of action
{
set(my,INVISIBLE | PASSABLE);<<<< set model invisible and passable
while(!zcar){wait(1);<<<<when pointer is not in game just wait
}
while(1)<<<<if pointer is in game
{
if (vec_dist (my.x, zcar.x) < 30){ <<<< when the distance between pointer and fire object is smaller than 30
snd_play(explo2_snd,150,0);<<<<play sound
ent_create("Explosion+16.tga", vector(zcar.x,zcar.y,zcar.z+20), sprite_played);<<<create explosion sprite at pointer coordinates
ent_create("damage.png", vector(my.x,my.y,my.z+50), score_up);<<<< create damage pane and give it the score up action
damage -=10;<<<< subtract 10 from damage variable
break;<<<< break off the action
}
effect(effect_flame, 1, my.x, nullvector); while none of above is taking place
wait(1);<<<<wait one frame
if (my.x < zcar.x-2000){<<<< when the model is behind the pointer 2000 or more
ent_remove(me);<<<<remove the model
break;<<<< break of the action
}
}
}
```



Line 887-911.

While driving the player can restore the damage taken by picking up jerrycans with fuel. Here is the action created for them. When player is in range the jerrycan will be picked up and disappear.

```
action gass_tank()<<<< name of action
{
set(my,PASSABLE | BRIGHT | METAL | SHADOW);<<<< set model passable bright shadow and give it metal look
while(!zcar){wait(1);<<<<When pointer is not in game just wait.
}
while(1)<<<<<while none of above
{
if (my.x < zcar.x-2000){<<<< when the model is behind the pointer 2000 or more
ent_remove(me);<<<<remove the model
break;<<<< break of the action
}
my.pan +=2*time_step;<<<<<make direction turn with speed of 2
my.tilt +=2*time_step;<<<<<make model tilt with speed of 2
if (vec_dist(zcar.x, my.x) < 30){<<<<<if distance between pointer and jerrycan is smaller then 30
damage +=25;<<<<<ad 25 to damage variable
ent_create("hundred.png", vector(my.x,my.y,my.z+50), <<<<<score_up); create score panel and give it score up function
snd_play(excel_snd,50,0);<<<<<play sound
ent_remove(me);<<<<remove model
score +=100;<<<<<ad 100 to score
break;<<<<<break of action
}
wait(1); <<<<<< wait one frame
}
```

Line 917-972.

Watch out ! wolves are running on the road. The use also emask and an event. They will start running when the player gets in range. Here is how:

First the impact function

```
function wolf_impact()<<<< name of the function
{
snd_play(carhit_snd,40,0);<<<< play sound
damage -=10;<<<< subtract 10 from damage variable
snd_play(wolf_snd,100,0);<<<<play sound
set(my,INVISIBLE | PASSABLE);<<<<set model invisible passable
reset(my,SHADOW);<<<<reset models shadow
ent_create("damage.png", vector(my.x,my.y,my.z+50), score_up);<<<<<create damage panel and give it score up function at models coordinates

ent_create("blood.png", vector(my.x,my.y,my.z-18), blood_pool);<<<<<create blood and give it blood pool function at models coordinates
ent_create("dwolf.mdl", vector(my.x,my.y-20,my.z-14), d_zombie);<<<<<create a new model with d zombie action at models coordinates
wait(-0.3);<<<<< wait minus 0.3 frames.
snd_play(blood_snd,50,0);<<<<play sound
ent_create("track.png", vector(zcar.x-50,zcar.y+10,zcar.z), track);<<<< create track at pointer coordinates
ent_create("track.png", vector(zcar.x-50,zcar.y-15,zcar.z), track);<<<< create track at pointer coordinates
wait(1);<<<< wait one frame
ent_remove(me);<<<<< remove the model
}
```




And then the wolf action.

```
action wolf()<<<<<name of the action
{
set(my,SHADOW);<<<< give model shadow
my.emask |= ENABLE_ENTITY | ENABLE_IMPACT;<<<<<emask used entity and impact
my.event= wolf_impact;<<<<<event starts wolf_impact function
var stand3_percentage;<<<<<create variable for stand animation
var run3_percentage;<<<<<create variable for run animation
c_setminmax(me);<<<<< Use real sie for collision
while(!zcar){wait(1);<<<<<if pointer is not in game just wait
}
while(1)<<<<<when pointer is in game
{
if(vec_dist (zcar.x, my.x) > 500){<<<<<if distance between pointer and wold is bigger then 500
stand3_percentage %= 100;<<<<<use 100% procent of animation stand
stand3_percentage += 2 * time_step;<<<<<play animation at speed 2
ent_animate(my, "stand", stand3_percentage, ANM_CYCLE);<<<<<<repeat animation
wait(1);<<<<< wait one frame
}
else<<<<<<if none of above
{
run3_percentage %= 100;<<<<<use 100% of animation run
run3_percentage += 10 * time_step;<<<<< play animation at speed 10
ent_animate(my, "run", run3_percentage, ANM_CYCLE);<<<<<<repeat animation

c_move(my, vector(10* time_step, 0, 0), nullvector, GLIDE | IGNORE_SPRITES | IGNORE_PASSABLE);<<<<< move from origin point at speed
10 x axis no movement on y and use glide ignore sprites and all passable models.
wait(1);<<<<<wait one frame
}
if (my.x < zcar.x-2000){<<<<< when the model is behind the pointer 2000 or more
ent_remove(me);<<<<<remove the model
break;<<<<< break of the action
}
}
```

Done ! A full game scripted and ready to be filled with new actions functions and visuals but that part is up to you. I think you have a good start with this template and I am looking forward to hear your reactions.



Some extra information.

This example template has only 2 levels. If you want to add more here is how to do it :

- *Open the level z-roads.*
- *Save this level now as stage003*
- *Built your level and save it again as stage 003*
- *Place zombies, wolves, fire objects and jerrykan models where you want and assign them there actions*

Now you only have to add this level in the level_switch function.

```
function level_switch()
{
    set(black_pan,SHOW | TRANSLUCENT);
    black_pan.alpha = 0;
    while(black_pan.alpha <100){
        black_pan.alpha +=5*time_step;
        wait(1);
    }
    reset(pan_zbonus,SHOW);
    reset(bonus_pan,SHOW);
    zbonus =0;
    stage +=1;
    wait(-1);
}
if(stage ==2){
    damage=279;
    game_enable = 0;
    level_load("stage002.WMB");
    ent_create("zcar01.mdl", vector(-150,0,-5), zombie_car);
}
if(stage ==3){
    damage=279;
    game_enable = 0;
    level_load("stage003.WMB");
    ent_create("zcar01.mdl", vector(-150,0,-5), zombie_car);
}

while(black_pan.alpha >0){
    black_pan.alpha -=5*time_step;
    wait(1);
}
begin_newlevel();
}
```

This way you can add as many levels as you want. Cool and easy isn't it ? ☺



Some need to know things about programming in lite_c

- *Scripts must end with ".c" (like "script01.c").*
- *Scripts must normally contain a function named main.*
- *A function contains several within a pair of winged brackets { }.*
- *A command has a list of one or more parameters that are placed within a pair of parentheses ().*
- *A function is defined using the keyword function followed by the name of the function and a pair of parentheses ()*
- *The body of the function (content) must be written inside a pair of curly brackets { }.*
- *The body of the function consists of one or more lines of lite-C code that end with a semicolon.*
- *The names used for the functions follow the same naming convention as for variables.*
- *You shouldn't use the same name for a variable and a function; this will lead to errors.*
- *A **string** is a sequence of characters: letters, numbers, symbols*
- *An action is just a function attached to an entity instead of function you use the word action*
- *You will need to use "if" instructions when you'll want your program to behave differently depending on some conditions*
- *The instructions placed inside the "else" part are executed only if "some condition" isn't true.*
- ***While** creates a loop. All in the while will repeat until the loop is broken*
- *take a look at the table to see how you can compare two different expressions*

SITUATION	OPERATOR	PRACTICAL EXAMPLE
expression1 = expression2	==	if (a == b)
expression1 is greater than expression2	>	if (a > b)
expression1 is greater than or equal to expression2	>=	if (a >= b)
expression1 is less than expression2	<	if (a < b)
expression1 is less than or equal to expression2	<=	if (a <= b)
expression1 does not equal expression2	!=	if (a != b)

Credits :

Sprites, Ingame music, Game concept, sound fx, Programming, Textures, 3Dmodels: René Pol.

3D Car model : Miquel A. Romar.

Score up and fire script : 3Run.

Programming tips and need to now : George Pirvu.